

CITY ENGINEERING COLLEGE

Approved by AICTE New Delhi & Affiliated by VTU, Belagavi

Doddakallasandra, Off Kanakapura Main Road, Next to Gokulam Apartment, Bangalore - 560 062.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(ACADEMIC YEAR 2025-26)

LABORATORY MANUAL

SUBJECT NAME : R PROGRAMMING LABORATORY

SUBJECT CODE: BCS358B

INSTITUTIONAL MISSION AND VISION

VISION

Making Remarkable Contribution by Disseminating Knowledge on Emerging Trends in Engineering and Technology through various Programs, Innovation and Research so as to excel in Quality both at National Level and International level and provide career guidance and training for Employment.

MISSION

M1: To encourage Knowledge Acquisition and Foster Innovation & Research

M2: To Prepare Students for Immediate Employment, leading to Technological and Social-Economic growth

M3: To Provide Guidance for a Productive Career under various programmes

PROGRAM OUTCOMES (Pos)

- PO1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and computer science and business systems to the solution of complex engineering and societal problems.
- PO2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering and business problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- PO3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- PO4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- PO5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- PO6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering and business practices.
- PO7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in business societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- PO8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering and business practices.
- PO9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- PO10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- PO11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering, business and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- PO12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Department of Computer Science & Engineering

Vision of the Department

To contribute to Global Development by producing Knowledgeable and Quality professionals who are Innovative and Successful in the advanced field of Computer Science & Engineering to adapt the changing Employment demands and social needs.

Mission of the Department

M1: To provide Quality Education to students, to build Confidence by developing their Technical Skills to make them Competitive Computer Science Engineers.

M2: To facilitate Innovation & Research for students and faculty and to provide Internship opportunities.

M3: To collaborate with educational institutions and industries for Excellence in Teaching and Research.

PROGRAM EDUCATION OUTCOMES (PEOs)

PEO1: Graduates will have strong foundation in Basic Engineering Sciences that are required for problem solving to succeed in their profession.

PEO2: Graduates will have scientific and engineering knowledge in cutting edge technologies so as to design and solve real life problems using the acquired skills and lifelong learning.

PEO3: Graduates will be professional with ethics, good communication skills, team work capability and relate engineering problems with social context.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: The Computer Science and Engineering graduates are able to analyze, design, develop, test and apply management principles, mathematical foundations in the development of computational solutions, make them to expert in designing the computer software and hardware.

PSO2: Develop their skills to solve problems in the broad area of programming concepts and appraise environmental and social issues with ethics and manage different projects in inter disciplinary field.

R Programming		Semester	3
Course Code	BCS358B	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Credits	01	Exam Hours	02
Examination type (SEE)	Practical		
Course objectives: <ul style="list-style-type: none">• To explore and understand how R and R Studio interactive environment.• To understand the different data Structures, data types in R.• To learn and practice programming techniques using R programming.• To import data into R from various data sources and generate visualizations.• To draw insights from datasets using data analytics techniques.			
SLNO	Experiments		
1	Demonstrate the steps for installation of R and R Studio. Perform the following: <ul style="list-style-type: none">a) Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.b) Demonstrate Arithmetic and Logical Operations with simple examples.c) Demonstrate generation of sequences and creation of vectors.d) Demonstrate Creation of Matricese) Demonstrate the Creation of Matrices from Vectors using Binding Function.f) Demonstrate element extraction from vectors, matrices and arrays Suggested Reading – Text Book 1 – Chapter 1 (What is R, Installing R, Choosing an IDE – RStudio, How to Get Help in R, Installing Extra Related Software), Chapter 2 (Mathematical Operations and Vectors, Assigning Variables, Special Numbers, Logical Vectors), Chapter 3 (Classes, Different Types of Numbers, Other Common Classes, Checking and Changing Classes, Examining Variables)		
2	Assess the Financial Statement of an Organization being supplied with 2 vectors of data: Monthly Revenue and Monthly Expenses for the Financial Year. You can create your own sample data vector for this experiment) Calculate the following financial metrics: <ul style="list-style-type: none">a. Profit for each month.b. Profit after tax for each month (Tax Rate is 30%).c. Profit margin for each month equals to profit after tax divided by revenue.d. Good Months – where the profit after tax was greater than the mean for the year.e. Bad Months – where the profit after tax was less than the mean for the year.f. The best month – where the profit after tax was max for the year.g. The worst month – where the profit after tax was min for the year. Note: <ul style="list-style-type: none">a. All Results need to be presented as vectorsb. Results for Dollar values need to be calculated with \$0.01 precision, but need to be presented inUnits of \$1000 (i.e 1k) with no decimal pointsc. Results for the profit margin ratio need to be presented in units of % with no decimal point.d. It is okay for tax to be negative for any given month (deferred tax asset)e. Generate CSV file for the data. Suggested Reading – Text Book 1 – Chapter 4 (Vectors, Combining Matrices)		
3	Develop a program to create two 3 X 3 matrices A and B and perform the following operations a) Transpose of the matrix b) addition c) subtraction d) multiplication Suggested Reading – Text Book 1 – Chapter 4 (Matrices and Arrays – Array Arithmetic)		
4	Develop a program to find the factorial of given number using recursive function calls. Suggested Reading – Reference Book 1 – Chapter 5 (5.5 – Recursive Programming) Text Book 1 – Chapter 8 (Flow Control and Loops – If and Else, Vectorized If, while loops, for loops),Chapter 6 (Creating and Calling Functions, Passing Functions to and from other functions)		

5	Develop an R Program using functions to find all the prime numbers up to a specified number by the method of Sieve of Eratosthenes. Suggested Reading – Reference Book 1 - Chapter 5 (5.5 – Recursive Programming) Text Book 1 – Chapter 8 (Flow Control and Loops – If and Else, Vectorized If, while loops, for loops), Chapter 6 (Creating and Calling Functions, Passing Functions to and from other functions)																				
6	The built-in data set mammals contain data on body weight versus brain weight. Develop R commands to: a) Find the Pearson and Spearman correlation coefficients. Are they similar? b) Plot the data using the plot command. c) Plot the logarithm (log) of each variable and see if that makes a difference. Suggested Reading – Text Book 1 – Chapter 12 – (Built-in Datasets) Chapter 14 – (Scatterplots) Reference Book 2 – 13.2.5 (Covariance and Correlation)																				
7	Develop R program to create a Data Frame with following details and do the following operations. <table><tr><th>itemCode</th><th>itemCategory</th><th>itemPrice</th></tr><tr><td>1001</td><td>Electronics</td><td>700</td></tr><tr><td>1002</td><td>Desktop Supplies</td><td>300</td></tr><tr><td>1003</td><td>Office Supplies</td><td>350</td></tr><tr><td>1004</td><td>USB</td><td>400</td></tr><tr><td>1005</td><td>CD Drive</td><td>800</td></tr></table> a) Subset the Data frame and display the details of only those items whose price is greater than or equal to 350. b) Subset the Data frame and display only the items where the category is either “Office Supplies” or “Desktop Supplies” c) Create another Data Frame called “item-details” with three different fields itemCode, ItemQtyonHand and ItemReorderLvl and merge the two frames Suggested Reading – Textbook 1: Chapter 5 (Lists and Data Frames)			itemCode	itemCategory	itemPrice	1001	Electronics	700	1002	Desktop Supplies	300	1003	Office Supplies	350	1004	USB	400	1005	CD Drive	800
itemCode	itemCategory	itemPrice																			
1001	Electronics	700																			
1002	Desktop Supplies	300																			
1003	Office Supplies	350																			
1004	USB	400																			
1005	CD Drive	800																			
8	Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973. Develop R program to generate histogram by using appropriate arguments for the following statements. a) Assigning names, using the air quality data set. b) Change colors of the Histogram c) Remove Axis and Add labels to Histogram d) Change Axis limits of a Histogram e) Add Density curve to the histogram Suggested Reading – Reference Book 2 – Chapter 7 (7.4 – The ggplot2 Package), Chapter 24 (Smoothing and Shading)																				
9	Design a data frame in R for storing about 20 employee details. Create a CSV file named “input.csv” that defines all the required information about the employee such as id, name, salary, start_date, dept. Import into R and do the following analysis. a) Find the total number rows & columns b) Find the maximum salary c) Retrieve the details of the employee with maximum salary d) Retrieve all the employees working in the IT Department. e) Retrieve the employees in the IT Department whose salary is greater than 20000 and write these																				

	<p>details into another file “output.csv”</p> <p>Suggested Reading – Text Book 1 – Chapter 12(CSV and Tab Delimited Files)</p>
10	<p>Using the built in dataset mtcars which is a popular dataset consisting of the design and fuel consumption patterns of 32 different automobiles. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). Format A data frame with 32 observations on 11 variables : [1] mpg Miles/(US) gallon, [2] cyl Number of cylinders [3] disp Displacement (cu.in.), [4] hp Gross horsepower [5] drat Rear axle ratio,[6] wt Weight (lb/1000) [7] qsec 1/4 mile time, [8] vs V/S, [9] am Transmission (0 = automatic, 1 = manual), [10] gear Number of forward gears, [11] carb Number of carburetors</p> <p>Develop R program, to solve the following:</p> <ol style="list-style-type: none"> What is the total number of observations and variables in the dataset? Find the car with the largest hp and the least hp using suitable functions Plot histogram/ density for each variable and determine whether continuous variables are normally distributed or not. If not, what is their skewness? What is the average difference of gross horse power(hp) between automobiles with 3 and 4 number of cylinders(cyl)? Also determine the difference in their standard deviations. Which pair of variables has the highest Pearson correlation? <p>References (Web links):</p> <ol style="list-style-type: none"> https://cran.r-project.org/web/packages/explore/vignettes/explore_mtcars.html https://www.w3schools.com/r/r_stat_data_set.asp https://rpubs.com/BillB/217355
11	<p>Demonstrate the progression of salary with years of experience using a suitable data set (You can create your own dataset). Plot the graph visualizing the best fit line on the plot of the given data points. Plot a curve of Actual Values vs. Predicted values to show their correlation and performance of the model.</p> <p>Interpret the meaning of the slope and y-intercept of the line with respect to the given data. Implement using lm function. Save the graphs and coefficients in files. Attach the predicted values of salaries as a new column to the original data set and save the data as a new CSV file.</p> <p>Suggested Reading – Reference Book 2 – Chapter 20 (General Concepts, Statistical Inference, Prediction)</p>

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.
- The marks scored shall be scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course are 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.
- The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.
- All laboratory experiments are to be included for practical examination.
(Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. OR based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva- voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.

- The minimum duration of SEE is 02 hours

Suggested Learning Resources:**Book:**

1. Cotton, R. (2013). Learning R: A Step by Step Function Guide to Data Analysis. 1st ed. O'Reilly Media Inc.

References:

1. Jones, O., Maillardet, R. and Robinson, A. (2014). Introduction to Scientific Programming and Simulation Using R. Chapman & Hall/CRC, The R Series.

Davies, T.M. (2016) The Book of R: A First Course in Programming and Statistics. No Starch Press.

R Programming Language

Introduction

R is an open-source programming language that is widely used as a statistical software and data analysis tool. R generally comes with the Command-line interface. R is available across widely used platforms like Windows, Linux, and macOS. Also, the R programming language is the latest cutting-edge tool.

It was designed by **Ross Ihaka and Robert Gentleman** at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team. R programming language is an implementation of the S programming language. It also combines with lexical scoping semantics inspired by Scheme. Moreover, the project conceives in 1992, with an initial version released in 1995 and a stable beta version in 2000.

Why R Programming Language?

- R programming is used as a leading tool for machine learning, statistics, and data analysis. Objects, functions, and packages can easily be created by R.
- It's a platform-independent language. This means it can be applied to all operating system.
- It's an open-source free language. That means anyone can install it in any organization without purchasing a license.
- R programming language is not only a statistic package but also allows us to integrate with other languages (C, C++). Thus, you can easily interact with many data sources and statistical packages.
- The R programming language has a vast community of users and it's growing day by day.
- R is currently one of the most requested programming languages in the Data Science job market that makes it the hottest trend nowadays.

Features of R Programming Language

Statistical Features of R:

- **Basic Statistics:** The most common basic statistics terms are the mean, mode, and median. These are all known as "Measures of Central Tendency." So using the R language we can measure central tendency very easily.
- **Static graphics:** R is rich with facilities for creating and developing interesting static graphics. R contains functionality for many plot types including graphic maps, mosaic plots, biplots, and the list goes on.

- **Probability distributions:** Probability distributions play a vital role in statistics and by using R we can easily handle various types of probability distribution such as Binomial Distribution, Normal Distribution, Chi-squared Distribution and many more.
- **Data analysis:** It provides a large, coherent and integrated collection of tools for data analysis.

Programming Features of R:

- **R Packages:** One of the major features of R is it has a wide availability of libraries. R has CRAN(Comprehensive R Archive Network), which is a repository holding more than 10, 0000 packages.
- **Distributed Computing:** Distributed computing is a model in which components of a software system are shared among multiple computers to improve efficiency and performance. Two new packages **ddR** and **multidplyr** used for distributed programming in R were released in November 2015.

Programming in R:

Since R is much similar to other widely used languages syntactically, it is easier to code and learn in R. Programs can be written in R in any of the widely used IDE like **R Studio**, **Rattle**, **Tinn-R**, etc. After writing the program save the file with the extension **.r**. To run the program use the following command on the command line:

R file_name.r

Example:

```
# R program to print Welcome to GFG!  
  
# Below line will print "Welcome to GFG!"  
cat("Welcome to GFG!")
```

Output:

Welcome to GFG!

What are R Data types?

R Data types are used in computer programming to specify the kind of data that can be stored in a variable. For effective memory consumption and precise computation, the right data type must be selected. Each R data type has its own set of regulations and restrictions.

Data Types in R Programming Language

Each variable in R has an associated data type. Each R-Data Type requires different amounts of memory and has some specific operations which can be performed over it. R Programming language has the following basic R-data types and the following table shows the data type and the values that each data type can take.

Basic Data Types	Values	Examples
Numeric	Set of all real numbers	"numeric_value <- 3.14"
Integer	Set of all integers, Z	"integer_value <- 42L"
Logical	TRUE and FALSE	"logical_value <- TRUE"
Complex	Set of complex numbers	"complex_value <- 1 + 2i"
Character	"a", "b", "c", ..., "@", "#", "\$", ..., "1", "2", ...etc	"character_value <- \"Hello Geeks\""
raw	as.raw()	"single_raw <- as.raw(255)"

Variables in R:

R Programming Language is a dynamically typed language, i.e. the R Language Variables are not declared with a data type rather they take the data type of the R-object assigned to them. This feature is also shown in languages like Python and PHP.

Declaring and Initializing Variables in R Language

R supports three ways of variable assignment:

- Using equal operator- operators use an arrow or an equal sign to assign values to variables.
- Using the leftward operator- data is copied from right to left.
- Using the rightward operator- data is copied from left to right.

R Variables Syntax

- Types of Variable Creation in R:
- Using equal to operators
variable_name = value
- using leftward operator
variable_name <- value
- using rightward operator
value -> variable_name

Taking Input from User in R Programming

Developers often have a need to interact with users, either to get data or to provide some sort of result. Most programs today use a dialog box as a way of asking the user to provide some type of input. Like other programming languages in R it's also possible to take input from the user. For doing so, there are two methods in R.

- Using **readline()** method
- Using **scan()** method

Using readline() method

In R language **readline()** method takes input in string format. If one inputs an integer then it is inputted as a string, lets say, one wants to input **255**, then it will input as **"255"**, like a string. So one needs to convert that inputted value to the format that he needs. In this case, string **"255"** is converted to integer 255. To convert the inputted value to the desired data type, there are some functions in R,

Data Structures in R Programming

A data structure is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks. Data structures in R programming are tools for holding multiple values.

R's base data structures are often organized by their dimensionality (1D, 2D, or nD) and whether they're homogeneous (all elements must be of the identical type) or heterogeneous (the elements are often of various types). This gives rise to the six data types which are most frequently utilized in data analysis.

The most essential data structures used in R include:

- **Vectors**
- **Lists**
- **Dataframes**
- **Matrices**
- **Arrays**
- **Factors**

Vectors

A vector is an ordered collection of basic data types of a given length. The only key thing here is all the elements of a vector must be of the identical data type e.g homogeneous data structures. Vectors are one-dimensional data structures.

Example:

```
# R program to illustrate Vector
# Vectors(ordered collection of same data type)
X = c(1, 3, 5, 7, 8)
# Printing those elements in console
print(X)
```

Output:

```
[1] 1 3 5 7 8
```

R Strings

Strings are a bunch of character variables. It is a one-dimensional array of characters. One or more characters enclosed in a pair of matching single or double quotes can be considered a string in R. Strings in R Programming represent textual content and can contain numbers, spaces, and special characters. An empty string is represented by using “”. R Strings are always stored as double-quoted values. A double-quoted string can contain single quotes within it. Single-quoted strings can't contain single quotes. Similarly, double quotes can't be surrounded by double quotes.

Creation of String in R

R Strings can be created by assigning character values to a variable. These strings can be further concatenated by using various functions and methods to form a big string.

Example

```
# R program for String Creation
# creating a string with double quotes
str1 <- "OK1"
cat ("String 1 is : ", str1)
# creating a string with single quotes
str2 <- 'OK2'
cat ("String 2 is : ", str2)
str3 <- "This is 'acceptable and 'allowed' in R"
cat ("String 3 is : ", str3)
str4 <- 'Hi, Wondering "if this "works"'
cat ("String 4 is : ", str4)
str5 <- 'hi, ' this is not allowed'
cat ("String 5 is : ", str5)
```

Output

String 1 is: OK1

String 2 is: OK2

String 3 is: This is 'acceptable and 'allowed' in R

String 4 is: Hi, Wondering "if this "works"

Error: unexpected symbol in " str5 <- 'hi, ' this"

Execution halted

R Vectors

R vectors are the same as the arrays in C language which are used to hold multiple data values of the same type. One major key point is that in R the indexing of the vector will start from '1' and not from '0'. We can create numeric vectors and character vectors as well.

Types of R vectors

Vectors are of different types which are used in R. Following are some of the types of vectors:

Numeric vectors: Numeric vectors are those which contain numeric values such as integer, float, etc.

```
# R program to create numeric Vectors
# creation of vectors using c() function.
v1<- c(4, 5, 6, 7)
# display type of vector
typeof(v1)
# by using 'L' we can specify that we want integer values.
v2<- c(1L, 4L, 2L, 5L)
# display type of vector
typeof(v2)
```

Output:

```
[1] "double"
```

```
[1] "integer"
```

Character vectors: Character vectors in R contain alphanumeric values and special characters.

```
# R program to create Character Vectors
# by default numeric values
# are converted into characters
v1<- c('geeks', '2', 'hello', 57)
# Displaying type of vector
typeof(v1)
```

Output:

```
[1] "character"
```

Logical vectors: Logical vectors in R contain Boolean values such as TRUE, FALSE and NA for Null values.

R

```
# R program to create Logical Vectors
# Creating logical vector
# using c() function
v1<- c(TRUE, FALSE, TRUE, NA)
# Displaying type of vector
typeof(v1)
```

Output:

```
[1] "logical"
```


Creating a vector

There are different ways of creating R vectors. Generally, we use 'c' to combine different elements together.

R

```
# R program to create Vectors
# we can use the c function
# to combine the values as a vector.
# By default the type will be double
X<- c(61, 4, 21, 67, 89, 2)
cat('using c function', X, '\n')
# seq() function for creating
# a sequence of continuous values.
# length.out defines the length of vector.
Y<- seq(1, 10, length.out = 5)
cat('using seq() function', Y, '\n')
# use ':' to create a vector
# of continuous values.
Z<- 2:7
cat('using colon', Z)
```

Output:

using c function 61 4 21 67 89 2

using seq() function 1 3.25 5.5 7.75 10

using colon 2 3 4 5 6 7

Length of R vector

R

```
# Create a numeric vector
x <- c(1, 2, 3, 4, 5)
# Find the length of the vector
length(x)
# Create a character vector
y <- c("apple", "banana", "cherry")
# Find the length of the vector
length(y)
# Create a logical vector
z <- c(TRUE, FALSE, TRUE, TRUE)
```

```
# Find the length of the vector  
length(z)
```

Output:

```
> length(x)
```

```
[1] 5
```

```
> length(y)
```

```
[1] 3
```

```
> length(z)
```

```
[1] 4
```

Accessing R vector elements

R – Lists

A list in R is a generic object consisting of an **ordered** collection of objects. Lists are one-dimensional, heterogeneous data structures. The list can be a list of vectors, a list of matrices, a list of characters and a list of functions, and so on.

A list is a vector but with heterogeneous data elements. A list in R is created with the use of **list()** function. R allows accessing elements of an R list with the use of the index value. In R, the indexing of a list starts with 1 instead of 0 like in other programming languages.

Creating a List

To create a List in R you need to use the function called “list()”. In other words, a list is a generic vector containing other objects. To illustrate how a list looks, we take an example here. We want to build a list of employees with the details. So for this, we want attributes such as ID, employee name, and the number of employees.

Example:

```
# R program to create a List  
# The first attributes is a numeric vector  
# containing the employee IDs which is created  
# using the command here  
  
empId = c(1, 2, 3, 4)
```

```
# The second attribute is the employee name

# which is created using this line of code here

# which is the character vector

empName = c("Debi", "Sandeep", "Subham", "Shiba")

# The third attribute is the number of employees

# which is a single numeric variable.

numberOfEmp = 4

# We can combine all these three different

# data types into a list

# containing the details of employees

# which can be done using a list command

empList = list(empId, empName, numberOfEmp)

print(empList)
```

Output:

```
[[1]]
```

```
[1] 1 2 3 4
```

```
[[2]]
```

```
[1] "Debi" "Sandeep" "Subham" "Shiba"
```

```
[[3]]
```

```
[1] 4
```

R – Array

Arrays are essential data storage structures defined by a fixed number of dimensions. Arrays are used for the allocation of space at contiguous memory locations. Uni-dimensional arrays are called vectors with the length being their only dimension. Two-dimensional arrays are called matrices, consisting of fixed numbers of rows and columns. Arrays consist of all elements of the same data type. Vectors are supplied as input to the function and then create an array based on the number of dimensions.

Creating an Array

An array in R can be created with the use of **array()** function. List of elements is passed to the array() functions along with the dimensions as required.

Syntax:

```
array(data, dim = (nrow, ncol, nmat), dimnames=names)
```

where,

nrow : Number of rows

ncol : Number of columns

nmat : Number of matrices of dimensions nrow * ncol

dimnames : Default value = NULL.

Otherwise, a list has to be specified which has a name for each component of the dimension. Each component is either a null or a vector of length equal to the dim value of that corresponding dimension.

Uni-Dimensional Array

A vector is a uni-dimensional array, which is specified by a single dimension, length. A Vector can be created using 'c()' function. A list of values is passed to the c() function to create a vector.

Example:

```
vec1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

print (vec1)

# cat is used to concatenate # strings and print it.

cat ("Length of vector : ", length(vec1))
```

Output:

```
[1] 1 2 3 4 5 6 7 8 9
```

```
Length of vector : 9
```

R – Matrices

Matrix is a rectangular arrangement of numbers in rows and columns. In a matrix, as we know rows are the ones that run horizontally and columns are the ones that run vertically. In R programming, matrices are two-dimensional, homogeneous data structures. These are some examples of matrices:

$$\begin{pmatrix} 1 & 5 & 3 \\ 4 & 9 & 2 \\ 5 & 6 & 7 \end{pmatrix} \quad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad [1 \ 4 \ 5]$$

R – Matrices**Creating Matrix**

To create a matrix in R you need to use the function called **matrix()**. The arguments to this **matrix()** are the set of elements in the vector. You have to pass how many numbers of rows and how many numbers of columns you want to have in your matrix.

Note: By default, matrices are in column-wise order.

R

```
# R program to create a matrix

A = matrix(

# Taking sequence of elements

c(1, 2, 3, 4, 5, 6, 7, 8, 9),

# No of rows

nrow = 3,
```

```
# No of columns

ncol = 3,

# By default matrices are in column-wise order

# So this parameter decides how to arrange the matrix

byrow = TRUE

)

# Naming rows

rownames(A) = c("a", "b", "c")

# Naming columns

colnames(A) = c("c", "d", "e")

cat("The 3x3 matrix:\n")

print(A)
```

Output:

The 3x3 matrix:

```
  c d e
a 1 2 3
b 4 5 6
c 7 8 9
```

R Factors

Factors in R Programming Language are data structures that are implemented to categorize the data or represent categorical data and store it on multiple levels.

They can be stored as integers with a corresponding label to every unique integer. The R factors may look similar to character vectors, they are integers and care must be taken while

using them as strings. The R factor accepts only a restricted number of distinct values. For example, a data field such as gender may contain values only from female, male, or transgender.

In the above example, all the possible cases are known beforehand and are predefined. These distinct values are known as levels. After a factor is created it only consists of levels that are by default sorted alphabetically.

Attributes of Factors in R Language

- **x:** It is the vector that needs to be converted into a factor.
- **Levels:** It is a set of distinct values which are given to the input vector x.
- **Labels:** It is a character vector corresponding to the number of labels.
- **Exclude:** This will mention all the values you want to exclude.
- **Ordered:** This logical attribute decides whether the levels are ordered.
- **nmax:** It will decide the upper limit for the maximum number of levels.

Creating a Factor in R Programming Language

The command used to create or modify a factor in R language is – **factor()** with a vector as input.

The two steps to creating an R factor :

- Creating a vector
- Converting the vector created into a factor using function factor()
-

Examples: Let us create a factor gender with levels female, male and transgender.

R

```
# Creating a vector

x <-c("female", "male", "male", "female")

print(x)

# Converting the vector x into a factor

# named gender

gender <-factor(x)

print(gender)
```

Output

```
[1] "female" "male" "male" "female"
```

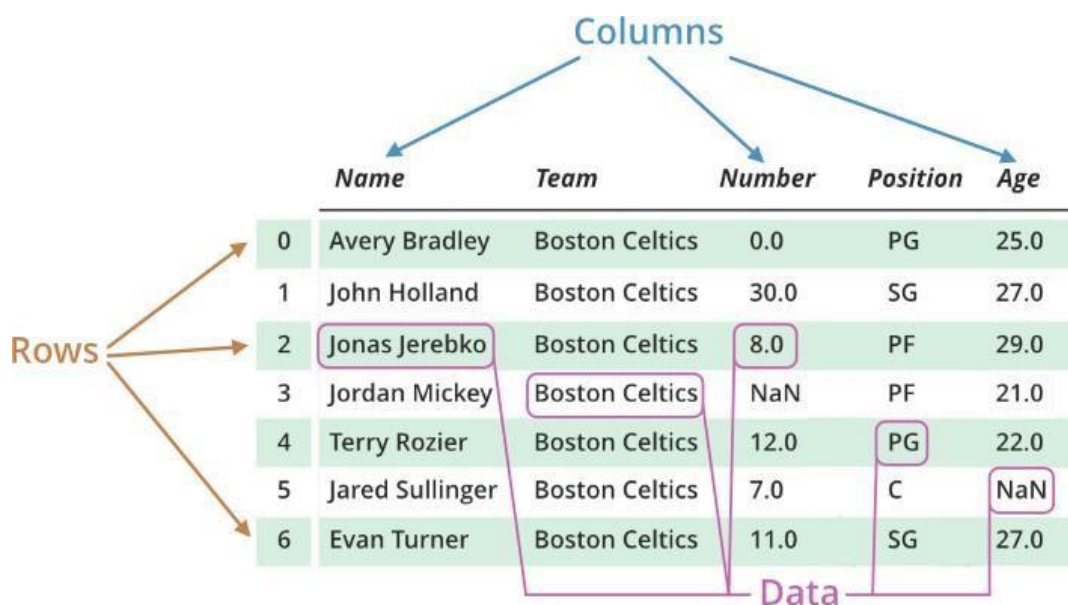
```
[1] female male male female
```

Levels: female male

R – Data Frames

R Programming Language is an open-source programming language that is widely used as a statistical software and data analysis tool. **Data Frames in R Language** are generic data objects of R that are used to store tabular data. Data frames can also be interpreted as matrices where each column of a matrix can be of different data types. R DataFrame is made up of three principal components, the data, rows, and columns.

R – Data Frames



R – Data Frames

Create Dataframe in R Programming Language

To create an R data frame use **data.frame()** command and then pass each of the vectors you have created as arguments to the function.

Example:**R**

```
# R program to create dataframe

# creating a data frame

friend.data <- data.frame(

  friend_id = c(1:5),

  friend_name = c("Sachin", "Sourav",

                  "Dravid", "Sehwag",

                  "Dhoni"),

  stringsAsFactors = FALSE

)

# print the data frame

print(friend.data)
```

Output:

```
  friend_id friend_name
1         1    Sachin
2         2    Sourav
3         3    Dravid
4         4    Sehwag
5         5     Dhoni
```

R dataset

A dataset is a data collection presented in a table.

The R programming language has tons of built-in datasets that can generally be used as a demo data to illustrate how the R functions work.

Most Used built-in Datasets in R

In R, there are tons of datasets we can try but the mostly used built-in datasets are:

airquality - New York Air Quality Measurements

AirPassengers - Monthly Airline Passenger Numbers 1949-1960

mtcars - Motor Trend Car Road Tests

iris - Edgar Anderson's Iris Data

.

PROGRAMS

Program 1:

Demonstrate the steps for installation of R and R Studio. Perform the following:

- a. Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.**
- b. Demonstrate Arithmetic and Logical Operations with simple examples.**
- c. Demonstrate generation of sequences and creation of vectors.**
- d. Demonstrate Creation of Matrices**
- e. Demonstrate the Creation of Matrices from Vectors using Binding Function.**
- f. Demonstrate element extraction from vectors, matrices and arrays**

Install R and RStudio

1. To install R, go to cran.r-project.org. ...
2. Click Download R for Windows.
3. Install R Click on install R for the first time.
4. Click Download R for Windows. ...
5. Select the language you would like to use during the installation. ...
6. Click Next.
7. Select where you would like R to be installed.

a. Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.

```
# -----  
# Assigning values of different data types  
# -----  
  
# Double (numeric with decimal)  
d <- 3.14159  
  
# Integer (use L for explicit integer type)  
i <- 42L  
  
# Logical (Boolean: TRUE/FALSE)  
l <- TRUE
```

```
# Complex number
cmp <- 5 + 2i

# Character (string / text)
c <- "Hello, R!"

# -----
# Display variable values and types
# -----
cat("d =", d, " | Type:", typeof(d), "\n")
cat("i =", i, " | Type:", typeof(i), "\n")
cat("l =", l, " | Type:", typeof(l), "\n")
cat("cmp =", cmp, " | Type:", typeof(cmp), "\n")
cat("c =", c, " | Type:", typeof(c), "\n")
```

OUTPUT:

```
> source("D:\\cec 3 rd sem\\New folder\\1b.R")
d = 3.14159 | Type: double
i = 42 | Type: integer
l = TRUE | Type: logical
cmp = 5+2i | Type: complex
c = Hello, R! | Type: character
```

B. Demonstrate Arithmetic and Logical Operations with simple examples.

```
# -----
# Arithmetic Operations
# -----

a <- 15
b <- 4

# Addition
add <- a + b

# Subtraction
sub <- a - b

# Multiplication
mul <- a * b

# Division
div <- a / b

# Integer Division (quotient only)
int_div <- a %/% b
```

```
# Modulus (remainder)
mod <- a %% b

# Exponentiation (power)
exp <- a ^ b

# Display Results
cat("a + b =", add, "\n")
cat("a - b =", sub, "\n")
cat("a * b =", mul, "\n")
cat("a / b =", div, "\n")
cat("a %/% b =", int_div, "\n")
cat("a %% b =", mod, "\n")
cat("a ^ b =", exp, "\n")
```

OUTPUT:

```
>
> source("D:/cec 3 rd sem/New folder/PROGRAMES/1B.R")
a + b = 19
a - b = 11
a * b = 60
a / b = 3.75
a %/% b = 3
a %% b = 3
a ^ b = 50625
```

C. Demonstrate generation of sequences and creation of vectors.

GENERATION OF SEQUENCES

```
# Using colon operator :
seq1 <- 1:10 # generates numbers from 1 to 10
print(seq1)

# Using seq() function
seq2 <- seq(1, 20, by = 2) # from 1 to 20 with step size 2
print(seq2)

# Generating sequence with length
seq3 <- seq(0, 1, length.out = 5) # 5 equally spaced numbers between 0 and 1
print(seq3)
```

OUTPUT:

```
> source("D:/cec 3 rd sem/New folder/PROGRAMES/1C.R")
[1] 1 2 3 4 5 6 7 8 9 10
[1] 1 3 5 7 9 11 13 15 17 19
[1] 0.00 0.25 0.50 0.75 1.00
> |
```

CREATION OF VECTORS

Numeric vector

```
num_vec <- c(10, 20, 30, 40, 50)
```

```
print(num_vec)
```

Character vector

```
char_vec <- c("R", "Python", "Java")
```

```
print(char_vec)
```

Logical vector

```
log_vec <- c(TRUE, FALSE, TRUE, TRUE)
```

```
print(log_vec)
```

Combining sequences into vector

```
combined_vec <- c(seq1, seq2)
```

```
print(combined_vec)
```

OUTPUT:

```
>
> source("D:/cec 3 rd sem/New folder/PROGRAMES/1D.R")
[1] 10 20 30 40 50
[1] "R"      "Python" "Java"
[1] TRUE FALSE TRUE TRUE
[1] 1 2 3 4 5 6 7 8 9 10 1 3 5 7 9 11 13 15 17 19
```

D. Demonstrate Creation of Matrices

Creating a matrix with numbers 1 to 9

```
mat1 <- matrix(1:9, nrow = 3, ncol = 3)
```

```
print("Matrix 1:")
```

```
print(mat1)
```

Filling by row instead of column

```
mat2 <- matrix(1:9, nrow = 3, byrow = TRUE)
```

```
print("Matrix 2 (filled by row):")
```

```
print(mat2)
```

OUTPUT:

```
> source("~/active-rstudio-document")
[1] "Matrix 1:"
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
[1] "Matrix 2 (filled by row):"
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> |
```

E. Demonstrate the Creation of Matrices from Vectors using Binding Function.

Creating two vectors

```
v1 <- c(1, 2, 3)
```

```
v2 <- c(4, 5, 6)
```

```
v3 <- c(7, 8, 9)
```

Column binding (cbind)

```
mat_cbind <- cbind(v1, v2, v3)
```

```
print("Matrix created using cbind:")
```

```
print(mat_cbind)
```

Row binding (rbind)

```
mat_rbind <- rbind(v1, v2, v3)
```

```
print("Matrix created using rbind:")
```

```
print(mat_rbind)
```

OUTPUT:

```
> source("~/active-rstudio-document")
[1] "Matrix created using cbind:"
      v1 v2 v3
[1,]  1  4  7
[2,]  2  5  8
[3,]  3  6  9
[1] "Matrix created using rbind:"
      [,1] [,2] [,3]
v1      1    2    3
v2      4    5    6
v3      7    8    9
> |
```

F . Demonstrate element extraction from vectors, matrices and arrays

Vector

```
vec <- c(10, 20, 30, 40, 50)
print("Vector:")
print(vec)
print(vec[2])    # 2nd element
print(vec[2:4])  # elements 2 to 4
```

Matrix

```
mat <- matrix(1:9, nrow = 3, byrow = TRUE)
print("Matrix:")
print(mat)
print(mat[2, 3]) # element at row 2, col 3
print(mat[2, ])  # entire 2nd row
print(mat[, 3])  # entire 3rd column
```

Array

```
arr <- array(1:24, dim = c(3, 4, 2))
print("Array:")
print(arr)
print(arr[2, 3, 1]) # element at row 2, col 3, layer 1
print(arr[2, , 1])  # 2nd row of 1st layer
```


OUTPUT:

```
> source("~/active-rstudio-document")
[1] "Vector:"
[1] 10 20 30 40 50
[1] 20
[1] 20 30 40
[1] "Matrix:"
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[1] 6
[1] 4 5 6
[1] 3 6 9
[1] "Array:"
, , 1

      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

      [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24

[1] 8
[1] 2 5 8 11
> |
```

Program 2:

Assess the Financial Statement of an Organization being supplied with 2 vectors of data: Monthly Revenue and Monthly Expenses for the Financial Year. You can create your own sample data vector for this experiment) Calculate the following financial metrics:

Profit for each month.

Profit after tax for each month (Tax Rate is 30%).

Profit margin for each month equals to profit after tax divided by revenue.

Good Months – where the profit after tax was greater than the mean for the year.

Bad Months – where the profit after tax was less than the mean for the year.

The best month – where the profit after tax was max for the year.

The worst month – where the profit after tax was min for the year.

PROGRAM:

```
# -----  
# Financial Statement Assessment  
# -----  
# Sample data for monthly revenue and expenses (in $1000 units)  
revenue <- c(50, 60, 55, 70, 65, 80, 75, 90, 85, 100, 95, 110)  
expenses <- c(30, 35, 33, 40, 38, 45, 42, 50, 48, 55, 52, 60)  
  
# Months  
months <- month.abb # Jan, Feb, Mar, ...  
  
# Profit = Revenue - Expenses  
profit <- revenue - expenses  
  
# Profit After Tax (30%)  
tax_rate <- 0.30  
profit_after_tax <- profit * (1 - tax_rate)  
  
# Profit Margin (%) = Profit After Tax / Revenue  
profit_margin <- profit_after_tax / revenue  
  
# Mean profit after tax for the year  
mean_pat <- mean(profit_after_tax)  
  
# Good Months (PAT > yearly average)  
good_months <- profit_after_tax > mean_pat
```

```
# Bad Months (PAT < yearly average)
bad_months <- profit_after_tax < mean_pat

# Best Month (max PAT)
best_month <- months[which.max(profit_after_tax)]

# Worst Month (min PAT)
worst_month <- months[which.min(profit_after_tax)]

# -----

# Results

# -----

df <- data.frame(
  Month = months,
  Revenue = revenue,
  Expenses = expenses,
  Profit = profit,
  Profit_After_Tax = profit_after_tax,
  Profit_Margin = round(profit_margin, 2),
  Good_Month = good_months,
  Bad_Month = bad_months
)

print(df)

cat("\nYearly Average Profit After Tax:", round(mean_pat, 2), "\n")
cat("Best Month:", best_month, "\n")
cat("Worst Month:", worst_month, "\n")
```

OUTPUT

```
> source("D:/cec 3 rd sem/New folder/exp2.R")
  Month Revenue Expenses Profit Profit_After_Tax Profit_Margin Good_Month Bad_Month
1   Jan      50       30    20          14.0         0.28      FALSE      TRUE
2   Feb      60       35    25          17.5         0.29      FALSE      TRUE
3   Mar      55       33    22          15.4         0.28      FALSE      TRUE
4   Apr      70       40    30          21.0         0.30      FALSE      TRUE
5   May      65       38    27          18.9         0.29      FALSE      TRUE
6   Jun      80       45    35          24.5         0.31       TRUE     FALSE
7   Jul      75       42    33          23.1         0.31      FALSE      TRUE
8   Aug      90       50    40          28.0         0.31       TRUE     FALSE
9   Sep      85       48    37          25.9         0.30       TRUE     FALSE
10  Oct     100       55    45          31.5         0.31       TRUE     FALSE
11  Nov      95       52    43          30.1         0.32       TRUE     FALSE
12  Dec     110       60    50          35.0         0.32       TRUE     FALSE

Yearly Average Profit After Tax: 23.74
Best Month: Dec
Worst Month: Jan
> |
```

Program 3:

Develop a program to create two 3 X 3 matrices A and B and perform the following operations

- a) Transpose of the matrix
- b) addition
- c) subtraction
- d) Multiplication

PROGRAM

```
# Create matrices A and B
A = matrix(1:9, nrow = 3, ncol = 3)
B = matrix(9:1, nrow = 3, ncol = 3)

# a) Transpose of the matrix
A_t = t(A)
B_t = t(B)

# b) Addition
sum = A + B
```

c) Subtraction

diff = A - B

d) Multiplication

prod = A %*% B

Print the results

cat("Matrix A:\n")

print(A)

cat("Matrix B:\n")

print(B)

cat("Transpose of A:\n")

print(A_t)

cat("Transpose of B:\n")

print(B_t)

cat("Addition of A and B:\n")

print(sum)

cat("Subtraction of A and B:\n")

print(diff)

cat("Multiplication of A and B:\n")

print(prod)

OUTPUT:

```
> source("~/active-rstudio-document")
Matrix A:
  [,1] [,2] [,3]
[1,]   1   4   7
[2,]   2   5   8
[3,]   3   6   9
Matrix B:
  [,1] [,2] [,3]
[1,]   9   6   3
[2,]   8   5   2
[3,]   7   4   1
Transpose of A:
  [,1] [,2] [,3]
[1,]   1   2   3
[2,]   4   5   6
[3,]   7   8   9
Transpose of B:
  [,1] [,2] [,3]
[1,]   9   8   7
[2,]   6   5   4
[3,]   3   2   1
Addition of A and B:
  [,1] [,2] [,3]
[1,]  10  10  10
[2,]  10  10  10
[3,]  10  10  10
```

Subtraction of A and B:

```
  [,1] [,2] [,3]
[1,]  -8  -2   4
[2,]  -6   0   6
[3,]  -4   2   8
```

Multiplication of A and B:

```
  [,1] [,2] [,3]
[1,]  90  54  18
[2,] 114  69  24
[3,] 138  84  30
```

Program 4:

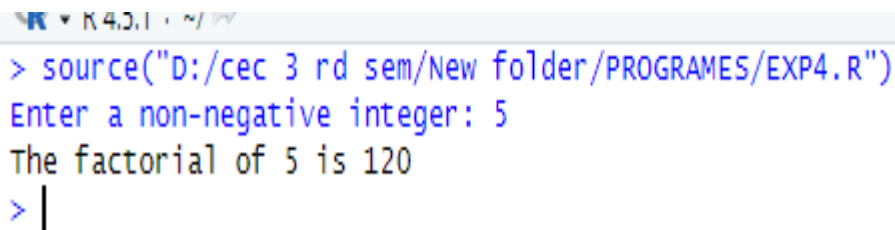
Develop a program to find the factorial of given number using recursive function calls.

```
# Recursive function to calculate the factorial

fact = function(n) {
  if (n == 0) {
    return(1)
  } else {
    return(n * fact(n - 1))
  }
}

# Input a number from the user
n = as.integer(readline("Enter a non-negative integer: "))
```

```
if (n < 0) {  
  cat("Factorial is not defined for negative numbers.\n")  
} else {  
  result = fact(n)  
  cat("The factorial of", n, "is", result, "\n")  
}
```

OUTPUT:

```
R 4.2.1 ~/  
> source("D:/cec 3 rd sem/New folder/PROGRAMES/EXP4.R")  
Enter a non-negative integer: 5  
The factorial of 5 is 120  
> |
```

Program 5:

Develop an R Program using functions to find all the prime numbers up to a specified number by the method of Sieve of Eratosthenes

```
# Function to find primes using Sieve of Eratosthenes
```

```
sieve_primes <- function(n) {  
  if (n < 2) {  
    return(NULL) # No primes less than 2  
  }  
}
```

```
# Step 1: Assume all numbers are prime
```

```
is_prime <- rep(TRUE, n)  
is_prime[1] <- FALSE # 1 is not prime
```

```
# Step 2: Eliminate multiples
```

```
for (i in 2:floor(sqrt(n))) {  
  if (is_prime[i]) {  
    is_prime[seq(i^2, n, by = i)] <- FALSE  
  }  
}
```

```
# Step 3: Return prime numbers
```

```
return(which(is_prime))  
}
```

```
# --- Dynamic Input ---
num <- as.integer(readline(prompt = "Enter a number: "))

# Find and display primes up to 'num'
cat("Prime numbers up to", num, "are:\n")
print(sieve_primes(num))
```

OUTPUT:

```
> source("D:/cec 3 rd sem/New folder/PROGRAMES/exp5.R")
Enter a number: 1
Prime numbers up to 1 are:
NULL

> source("D:/cec 3 rd sem/New folder/PROGRAMES/exp5.R")
Enter a number: 30
Prime numbers up to 30 are:
[1]  2  3  5  7 11 13 17 19 23 29
>
```

Program 6:

The built-in data set mammals contain data on body weight versus brain weight. Develop R commands to:

- Find the Pearson and Spearman correlation coefficients. Are they similar?
- Plot the data using the plot command.
- Plot the logarithm (log) of each variable and see if that makes a difference.

STEP 1: Load the Dataset

```
# Load MASS package (contains mammals dataset)
library(MASS)

# View first few rows
head(mammals)
```

#a) Find the Pearson and Spearman correlation coefficients. Are they similar?

```
# Pearson correlation
pearson_corr <- cor(mammals$body, mammals$brain, method = "pearson")

# Spearman correlation
spearman_corr <- cor(mammals$body, mammals$brain, method = "spearman")
```



```
# Print results
```

```
cat("Pearson Correlation:", pearson_corr, "\n")
```

```
cat("Spearman Correlation:", spearman_corr, "\n")
```

#b) Plot the data using the plot command.

```
plot(mammals$body, mammals$brain,
```

```
  xlab = "Body Weight",
```

```
  ylab = "Brain Weight",
```

```
  main = "Mammals: Body Weight vs Brain Weight",
```

```
  pch = 19, col = "blue")
```

#c) Plot the logarithm (log) of each variable and see if that makes a difference.

```
plot(log(mammals$body), log(mammals$brain),
```

```
  xlab = "log(Body Weight)",
```

```
  ylab = "log(Brain Weight)",
```

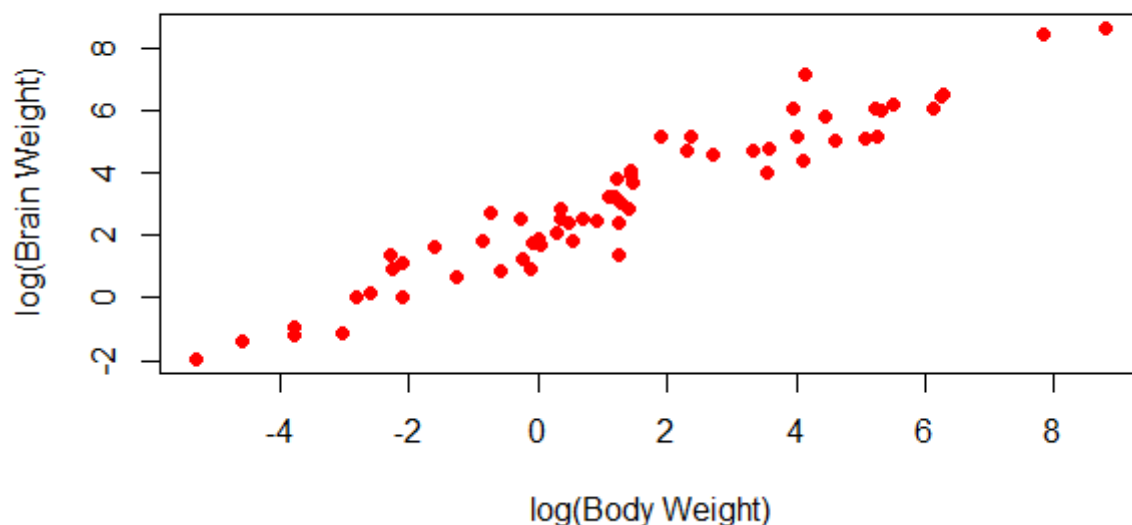
```
  main = "Log-Log Plot: Body Weight vs Brain Weight",
```

```
  pch = 19, col = "red")
```

OUTPUT:

```
> source("~/active-rstudio-document")
Pearson Correlation: 0.9341638
Spearman Correlation: 0.9534986
```

Log-Log Plot: Body Weight vs Brain Weight



Program 7:

Develop R program to create a Data Frame with following details and do the following operations.

itemCode	itemCategory
1001	Electronics
1002	Desktop Supplies
1003	Office Supplies
1004	USB
1005	CD Drive

Subset the Data frame and display the details of only those items whose price is greater than or equal to 350.

Subset the Data frame and display only the items where the category is either “Office Supplies” or

“Desktop Supplies”

Create another Data Frame called “item-details” with three different fields itemCode, ItemQtyonHand and ItemReorderLvl and merge the two frames

```
# Create Data Frame with item details
items <- data.frame(
  itemCode = c(1001, 1002, 1003, 1004, 1005),
  itemCategory = c("Electronics", "Desktop Supplies", "Office Supplies", "USB", "CD Drive"),
  price = c(15000, 500, 300, 2000, 350) # Added price column
)

cat("Original Items Data Frame:\n")
print(items)
# 1. Subset items with price >= 350
high_price_items <- subset(items, price >= 350)
cat("\nItems with price >= 350:\n")
print(high_price_items)

# 2. Subset items in category Office Supplies or Desktop Supplies
supply_items <- subset(items, itemCategory %in% c("Office Supplies", "Desktop Supplies"))
cat("\nItems in Office Supplies or Desktop Supplies:\n")
print(supply_items)
```

3. Create another Data Frame: item-details

```
item_details <- data.frame(  
  itemCode = c(1001, 1002, 1003, 1004, 1005),  
  ItemQtyonHand = c(10, 50, 200, 15, 30),  
  ItemReorderLvl = c(2, 10, 50, 5, 8)  
)
```

```
cat("\nItem Details Data Frame:\n")  
print(item_details)
```

4. Merge the two Data Frames on itemCode

```
merged_data <- merge(items, item_details, by = "itemCode")  
cat("\nMerged Data Frame:\n")  
print(merged_data)
```

OUTPUT:

```
> source("~/active-rstudio-document")  
Original Items Data Frame:  
  itemCode  itemCategory price  
1    1001    Electronics 15000  
2    1002 Desktop Supplies   500  
3    1003 office Supplies   300  
4    1004             USB   2000  
5    1005             CD Drive   350  
  
Items with price >= 350:  
  itemCode  itemCategory price  
1    1001    Electronics 15000  
2    1002 Desktop Supplies   500  
4    1004             USB   2000  
5    1005             CD Drive   350  
  
Items in Office Supplies or Desktop Supplies:  
  itemCode  itemCategory price  
2    1002 Desktop Supplies   500  
3    1003 office Supplies   300  
  
Item Details Data Frame:  
  itemCode ItemQtyonHand ItemReorderLvl  
1    1001             10             2  
2    1002             50            10  
3    1003            200            50  
4    1004             15             5  
5    1005             30             8  
  
Merged Data Frame:  
  itemCode  itemCategory price ItemQtyonHand ItemReorderLvl  
1    1001    Electronics 15000             10             2  
2    1002 Desktop Supplies   500             50            10  
3    1003 office Supplies   300            200            50  
4    1004             USB   2000             15             5  
5    1005             CD Drive   350             30             8  
> |
```

Program 8:

Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973. Develop R program to generate histogram by using appropriate arguments for the following statements.

- a) Assigning names, using the air quality data set.
- b) Change colors of the Histogram
- c) Remove Axis and Add labels to Histogram
- d) Change Axis limits of a Histogram
- e) Add Density curve to the histogram.

```
# Load built-in dataset
```

```
data("airquality")
```

```
# Assign proper names (already there, but we reassign for clarity)
```

```
names(airquality) <- c("Ozone", "Solar.R", "Wind", "Temp", "Month", "Day")
```

```
# Take Ozone column and remove NA values
```

```
ozone_data <- na.omit(airquality$Ozone)
```

```
# a) Basic Histogram
```

```
hist(ozone_data,
```

```
  main = "Histogram of Ozone",
```

```
  xlab = "Ozone Levels")
```

```
# b) Change colors of the Histogram
```

```
hist(ozone_data,
```

```
  main = "Colored Histogram",
```

```
xlab = "Ozone Levels",  
col = "lightblue", # fill color  
border = "black") # border color
```

c) Remove Axis and Add labels

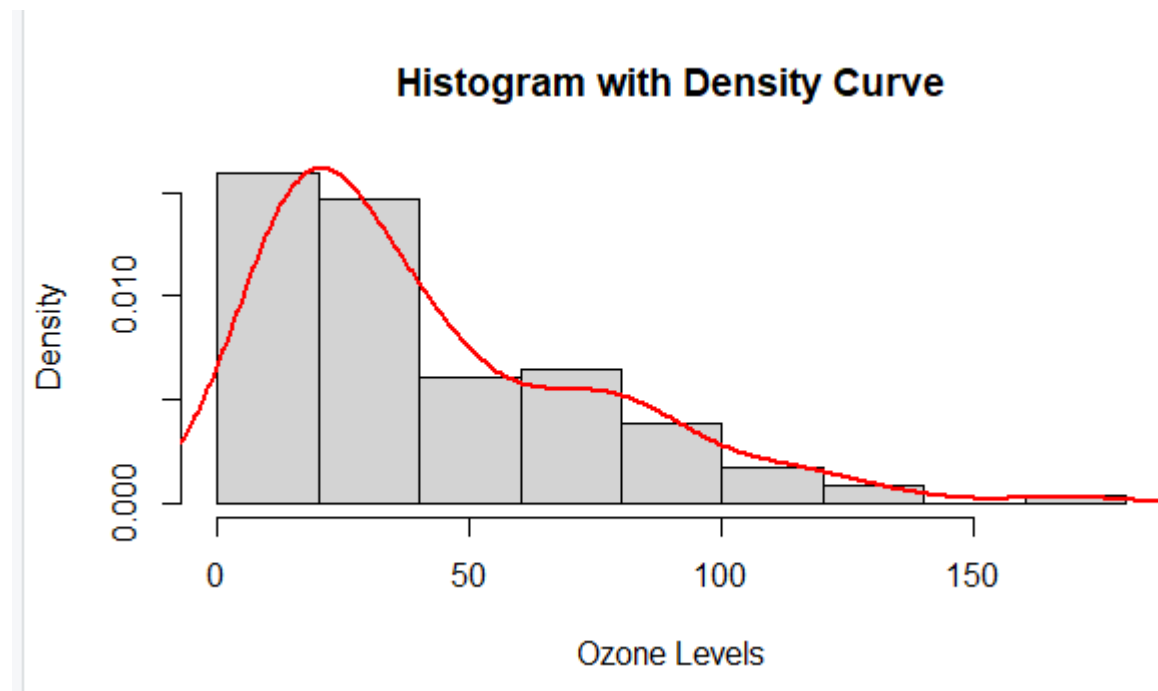
```
hist(ozone_data,  
     main = "Histogram without Axis",  
     xlab = "Ozone Levels",  
     ylab = "Frequency",  
     axes = FALSE, # removes axes  
     col = "orange")
```

d) Change Axis limits

```
hist(ozone_data,  
     main = "Histogram with Axis Limits",  
     xlab = "Ozone Levels",  
     col = "green",  
     xlim = c(0, 200), # X-axis range  
     ylim = c(0, 50)) # Y-axis range
```

e) Add Density Curve

```
hist(ozone_data,  
     main = "Histogram with Density Curve",  
     xlab = "Ozone Levels",  
     col = "lightgray",  
     freq = FALSE) # histogram in density scale  
lines(density(ozone_data),  
      col = "red",  
      lwd = 2) # add red density curve
```

OUTPUT:**Program 9:**

Design a data frame in R for storing about 20 employee details. Create a CSV file named “input.csv” that defines all the required information about the employee such as id, name, salary, start_date, dept. Import into R and do the following analysis.

- Find the total number rows & columns
- Find the maximum salary
- Retrieve the details of the employee with maximum salary
- Retrieve all the employees working in the IT Department.
- Retrieve the employees in the IT Department whose salary is greater than 20000 and write these details into another file “output.csv”

STEP 1: Create a CSV file and write a employee data

```
# -----  
# Step 1: Create Employee Data Frame  
# -----  
employee <- data.frame(  
  id = 1:20,  
  name = c("John","Alice","Bob","David","Eva","Frank","Grace","Hannah","Ian","Jack",  
           "Kiran","Lily","Mike","Nina","Oscar","Priya","Quinn","Ravi","Sophia","Tom"),  
  salary = c(18000,22000,25000,30000,15000,28000,32000,21000,26000,24000,  
            23000,27000,35000,19000,40000,17000,22000,31000,33000,20000),
```

```
start_date = as.Date(c("2015-01-10","2016-03-15","2014-07-20","2017-06-01","2018-08-12",
                      "2013-11-25","2019-02-18","2020-05-05","2012-09-14","2016-10-10",
                      "2017-12-22","2014-04-09","2011-03-19","2015-07-27","2010-01-30",
                      "2021-09-12","2018-06-11","2013-08-08","2012-12-02","2019-11-15")),
dept = c("HR","IT","Finance","IT","HR","Finance","IT","Finance","IT","HR",
         "Finance","IT","Finance","HR","IT","Finance","HR","IT","Finance","HR")
)

# Save to CSV file
write.csv(employee, "C:/Users/Admin/Documents/input.csv", row.names = FALSE)
```

STEP 2: Read the date in input.csv file

```
# -----
# Step 2: Import CSV File
# -----
emp_data <- read.csv("C:/Users/Admin/Documents/input.csv")

# a) Find total number of rows & columns
cat("Number of Rows:", nrow(emp_data), "\n")
cat("Number of Columns:", ncol(emp_data), "\n")

# b) Find maximum salary
max_salary <- max(emp_data$salary)
cat("Maximum Salary:", max_salary, "\n")

# c) Retrieve details of employee with maximum salary
cat("Employee with Maximum Salary:\n")
print(subset(emp_data, salary == max_salary))

# d) Retrieve employees in IT Department
cat("Employees in IT Department:\n")
print(subset(emp_data, dept == "IT"))

# e) Retrieve employees in IT Department with salary > 20000
it_high_salary <- subset(emp_data, dept == "IT" & salary > 20000)
cat("IT Employees with Salary > 20000:\n")
print(it_high_salary)

# Write these details into another CSV file
write.csv(it_high_salary, "output.csv", row.names = FALSE)
```

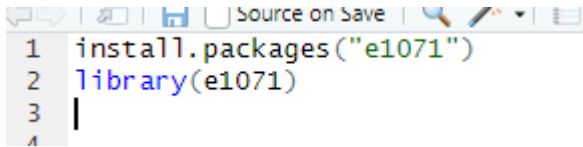
OUTPUT:

```
> source("~/active-rstudio-document")
Number of Rows: 20
Number of Columns: 5
Maximum Salary: 40000
Employee with Maximum Salary:
  id name salary start_date dept
15 15 Oscar 40000 2010-01-30 IT
Employees in IT Department:
  id name salary start_date dept
2 2 Alice 22000 2016-03-15 IT
4 4 David 30000 2017-06-01 IT
7 7 Grace 32000 2019-02-18 IT
9 9 Ian 26000 2012-09-14 IT
12 12 Lily 27000 2014-04-09 IT
15 15 Oscar 40000 2010-01-30 IT
18 18 Ravi 31000 2013-08-08 IT
IT Employees with Salary > 20000:
  id name salary start_date dept
2 2 Alice 22000 2016-03-15 IT
4 4 David 30000 2017-06-01 IT
7 7 Grace 32000 2019-02-18 IT
9 9 Ian 26000 2012-09-14 IT
12 12 Lily 27000 2014-04-09 IT
15 15 Oscar 40000 2010-01-30 IT
18 18 Ravi 31000 2013-08-08 IT
> |
```

Program 10 :

Using the built in dataset mtcars which is a popular dataset consisting of the design and fuel consumption patterns of 32 different automobiles. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). Format A data frame with 32 observations on 11 variables : [1] mpg Miles/(US) gallon, [2] cyl Number of cylinders [3] disp Displacement (cu.in.), [4] hp Gross horsepower [5] drat Rear axle ratio,[6] wt Weight (lb/1000) [7] qsec 1/4 mile time, [8] vs V/S, [9] am Transmission (0 = automatic, 1 = manual), [10] gear Number of forward gears, [11] carb Number of carburetors. Develop R program, to solve the following

- What is the total number of observations and variables in the dataset?
- Find the car with the largest hp and the least hp using suitable functions
- Plot histogram / density for each variable and determine whether continuous variables are normally distributed or not. If not, what is their skewness?
- What is the average difference of gross horse power(hp) between automobiles with 3 and 4 number of cylinders(cyl)? Also determine the difference in their standard deviations.
- Which pair of variables has the highest Pearson correlation?

STEP1: Install the Library (e1071)

```
1 install.packages("e1071")
2 library(e1071)
3 |
4
```

STEP2:

```
# Load the built-in dataset
data("mtcars")

# -----
# a) Total number of observations and variables
# -----
cat("Number of Observations (rows):", nrow(mtcars), "\n")
cat("Number of Variables (columns):", ncol(mtcars), "\n")

# -----
# b) Car with largest and least horsepower
# -----
max_hp <- max(mtcars$hp)
min_hp <- min(mtcars$hp)

cat("\nCar with Maximum Horsepower:\n")
print(mtcars[mtcars$hp == max_hp, ])

cat("\nCar with Minimum Horsepower:\n")
print(mtcars[mtcars$hp == min_hp, ])

# -----
# c) Histograms + Density + Skewness
# -----
par(mfrow = c(3, 4)) # show multiple plots together
for (var in names(mtcars)) {
  hist(mtcars[[var]],
       main = paste("Histogram of", var),
       xlab = var,
       col = "lightblue",
       border = "black",
       freq = FALSE)
  lines(density(mtcars[[var]]), col = "red", lwd = 2)
}
par(mfrow = c(1, 1)) # reset layout

# Skewness check
library(e1071) # provides skewness() function
cat("\nSkewness of variables:\n")
for (var in names(mtcars)) {
  sk <- skewness(mtcars[[var]])
```

```

cat(var, ":", sk, "\n")
}

# -----
# d) HP difference between cars with 3 and 4 cylinders
# -----
hp_cyl3 <- mtcars$hp[mtcars$cyl == 3]
hp_cyl4 <- mtcars$hp[mtcars$cyl == 4]

mean_diff <- mean(hp_cyl3) - mean(hp_cyl4)
sd_diff <- sd(hp_cyl3) - sd(hp_cyl4)

cat("\nAverage HP difference (3 cyl - 4 cyl):", mean_diff, "\n")
cat("SD difference of HP (3 cyl - 4 cyl):", sd_diff, "\n")

# -----
# e) Pair of variables with highest Pearson correlation
# -----
cor_matrix <- cor(mtcars) # correlation matrix
max_corr <- max(cor_matrix[upper.tri(cor_matrix)])
where <- which(cor_matrix == max_corr, arr.ind = TRUE)

var1 <- rownames(cor_matrix)[where[1]]
var2 <- colnames(cor_matrix)[where[2]]

cat("\nHighest Pearson Correlation:\n")
cat(var1, "and", var2, "with correlation =", max_corr, "\n")

```

OUTPUT:

```

>
> source("~/active-rstudio-document")
Number of Observations (rows): 32
Number of Variables (columns): 11

Car with Maximum Horsepower:
      mpg cyl disp  hp drat   wt  qsec vs am gear carb
Maserati Bora  15   8  301 335 3.54 3.57 14.6  0  1   5   8

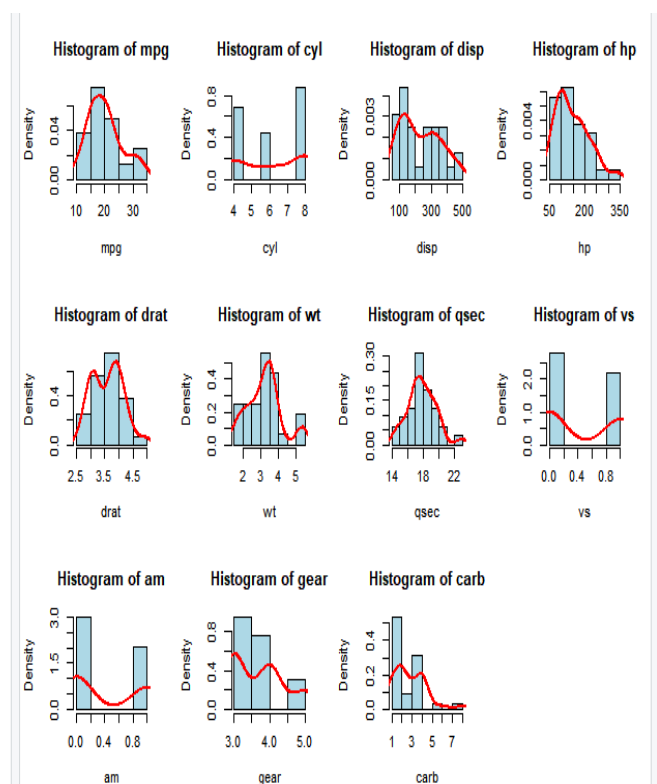
Car with Minimum Horsepower:
      mpg cyl disp  hp drat   wt  qsec vs am gear carb
Honda Civic  30.4  4  75.7 52 4.93 1.615 18.52  1  1   4   2

Skewness of variables:
mpg : 0.610655
cyl : -0.1746119
disp : 0.381657
hp : 0.7260237
drat : 0.2659039
wt : 0.4231465
qsec : 0.3690453
vs : 0.2402577
am : 0.3640159
gear : 0.5288545
carb : 1.050874

Average HP difference (3 cyl - 4 cyl): NaN
SD difference of HP (3 cyl - 4 cyl): NA

Highest Pearson Correlation:
disp and cyl with correlation = 0.9020329
>

```



Program 11:

Demonstrate the progression of salary with years of experience using a suitable data set (You can create your own dataset). Plot the graph visualizing the best fit line on the plot of the given data points. Plot a curve of Actual Values vs. Predicted values to show their correlation and performance of the model.

Interpret the meaning of the slope and y-intercept of the line with respect to the given data. Implement using lm function. Save the graphs and coefficients in files. Attach the predicted values of salaries as a new column to the original data set and save the data as a new CSV file.

```
# -----  
# Linear Regression in R  
# Salary vs Experience Example  
# -----  
  
# Step 1: Create a dataset  
experience <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
salary <- c(25000, 28000, 32000, 35000, 39000,  
            42000, 46000, 49000, 52000, 56000)  
  
employee_data <- data.frame(experience, salary)  
  
# Step 2: Fit linear regression model  
model <- lm(salary ~ experience, data = employee_data)  
  
# Step 3: Summary of model  
summary(model)  
  
# Step 4: Plot scatter with best-fit regression line  
png("salary_regression.png") # save graph to file  
plot(employee_data$experience, employee_data$salary,  
      main = "Salary vs Experience",  
      xlab = "Years of Experience", ylab = "Salary",  
      pch = 19, col = "blue")  
abline(model, col = "red", lwd = 2) # regression line  
dev.off()  
  
# Step 5: Predicted values  
employee_data$predicted_salary <- predict(model, newdata = employee_data)  
  
# Step 6: Plot Actual vs Predicted values  
png("actual_vs_predicted.png")  
plot(employee_data$salary, employee_data$predicted_salary,  
      main = "Actual vs Predicted Salaries",  
      xlab = "Actual Salary", ylab = "Predicted Salary",  
      pch = 19, col = "darkgreen")
```

```
abline(0, 1, col = "red", lwd = 2) # ideal correlation line
dev.off()
```

```
# Step 7: Save regression coefficients
```

```
coeffs <- coef(model)
```

```
write.csv(as.data.frame(coeffs), "coefficients.csv", row.names = TRUE)
```

```
# Step 8: Save the dataset with predictions
```

```
write.csv(employee_data, "employee_salary_predictions.csv", row.names = FALSE)
```

```
# -----
```

```
# Interpretation:
```

```
# slope = coeffs["experience"]
```

```
# intercept = coeffs["(Intercept)"]
```

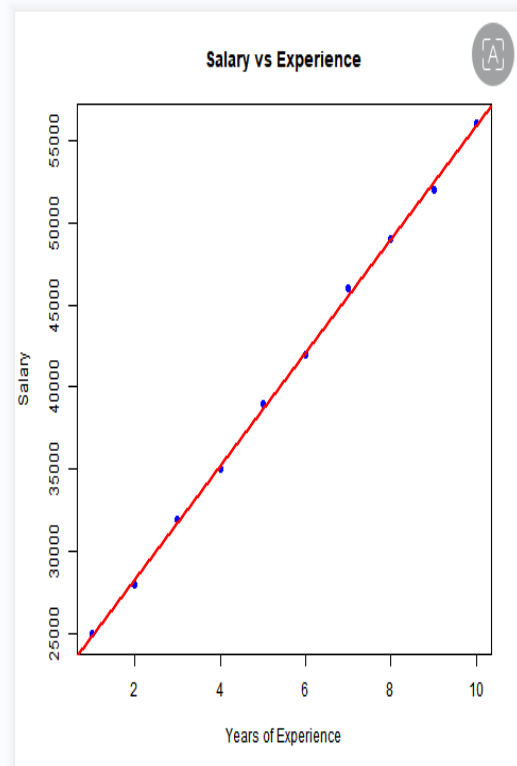
```
# slope means: Increase in salary per 1 year of experience
```

```
# intercept means: Base salary when experience = 0
```

```
# -----
```

OUTPUT:

	A	B	C	D	E	F
1	experience	salary	predicted_salary			
2	1	25000	24909.0909090909			
3	2	28000	28351.5151515152			
4	3	32000	31793.9393939394			
5	4	35000	35236.3636363636			
5	5	39000	38678.7878787879			
7	6	42000	42121.2121212121			
3	7	46000	45563.6363636364			
9	8	49000	49006.0606060606			
0	9	52000	52448.4848484848			
1	10	56000	55890.9090909091			
2						
3						
4						
5						
6						



VIVA QUATION AND ANSWERS**1. Explain what is R?**

R is data analysis software which is used by analysts, quants, statisticians, data scientists and others.

2. What is R programming and what are main feature of R?

R Programming Language is an open-source programming language that is widely used as a statistical software and data analysis tool. R generally comes with the Command-line interface. R is available across widely used platforms like Windows, Linux, and macOS. Also, the R programming language is the latest cutting-edge tool.

R programming is used as a leading tool for machine learning, statistics, and data analysis. Objects, functions, and packages can easily be created by R.

The main features of R:

Open source

Extensible

Statistical computing

Reproducibility

Graphics

3. List out some of the function that R provides?

The function that R provides are

- Mean
- Median
- Distribution
- Covariance
- Regression
- Non-linear
- Mixed Effects
- GLM
- GAM. etc.

4. Explain how you can start the R commander GUI?

Typing the command, ("Rcmdr") into the R console starts the R commander GUI.

5. In R how you can import Data?

You use R commander to import Data in R, and there are three ways through which you can enter data into it

You can enter data directly via Data ☐ New Data Set

Import data from a plain text (ASCII) or other files (SPSS, Minitab, etc.)

Read a data set either by typing the name of the data set or selecting the data set in the dialog box

6. Mention what does not 'R' language do?

Though R programming can easily connects to DBMS is not a database

R does not consist of any graphical user interface

Though it connects to Excel/Microsoft Office easily, R language does not provide any spreadsheet view of data.

7. Explain how R commands are written?

In R, anywhere in the program you have to preface the line of code with a #sign, for example

subtraction

division

note order of operations exists

8. How can you save your data in R?

To save data in R, there are many ways, but the easiest way of doing this is

Go to Data > Active Data Set > Export Active Data Set and a dialogue box will appear, when you click ok the dialogue box let you save your data in the usual way.

9. Mention how you can produce co-relations and covariances?

You can produce co-relations by the cor () function to produce co-relations and cov () function to produce covariances.

10. Explain what is t-tests in R?

In R, the t.test () function produces a variety of t-tests. T-test is the most common test in statistics and used to determine whether the means of two groups are equal to each other.

11. What are the data structures in R that is used to perform statistical analyses and create graphs?

R has data structures like

Vectors

Matrices

Arrays

Data frames

12. What is a data frame?

A data frame is made up of rows and columns, where each row denotes an observation or record and each column a variable or attribute. A data frame's columns can include a variety of data kinds, including logical, character, factor, and numeric ones, enabling the storing and management of the data.

13. How to create a data frame in R?

Use the `data.frame()` function in R to build a data frame. A two-dimensional tabular data structure called a data frame divides data into rows and columns. A data frame's columns can each contain a different data type, such as a logical, character, factor, or numeric one.

14. What is the function used for adding datasets in R?

`rbind` function can be used to join two data frames (datasets). The two data frames must have the same variables, but they do not have to be in the same order.

15. Definitions

- A vector is simply a list of items that are of the same type.
- A list in R can contain many different data types inside it. A list is a collection of data which is ordered and changeable.
To create a list, use the `list()` function
- A matrix is a two dimensional data set with columns and rows.
A column is a vertical representation of data, while a row is a horizontal representation of data.
A matrix can be created with the `matrix()` function. Specify the `nrow` and `ncol` parameters to get the amount of rows and columns
- Compared to matrices, arrays can have more than two dimensions.
We can use the `array()` function to create an array, and the `dim` parameter to specify the dimensions
- Data Frames are data displayed in a format as a table.
- Data Frames can have different types of data inside it. While the first column can be character, the second and third can be numeric or logical. However, each column should have the same type of data.
Use the `data.frame()` function to create a data frame: